INTRODUCCIÓN A CUDA

X JORNADAS DE DIVULGACIÓN DE APLICACIONES CIENTÍFICAS Y VISIÓN POR COMPUTADOR SOBRE PROCESADORES GRÁFICOS

UNIVERSIDAD DE ALICANTE

Pablo Martínez-González <pmartinez@dtic.ua.es>

Albert García-García <agarcia@dtic.ua.es>

José García-Rodríguez <jgarcia@dtic.ua.es>

CONTENIDO

¿QUÉ ES CUDA?

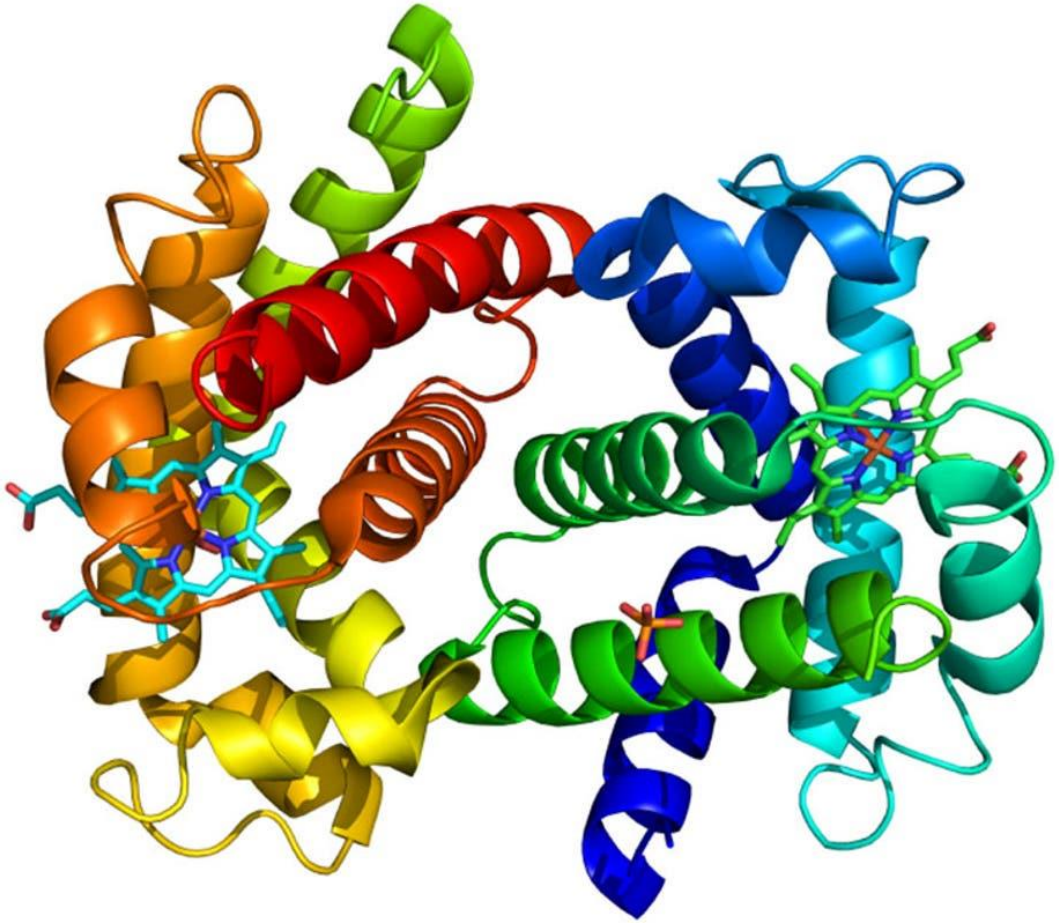
NVIDIA RTX 3090 (2020)

¿QUÉ ES CUDA?

VIDEOJUEGOS

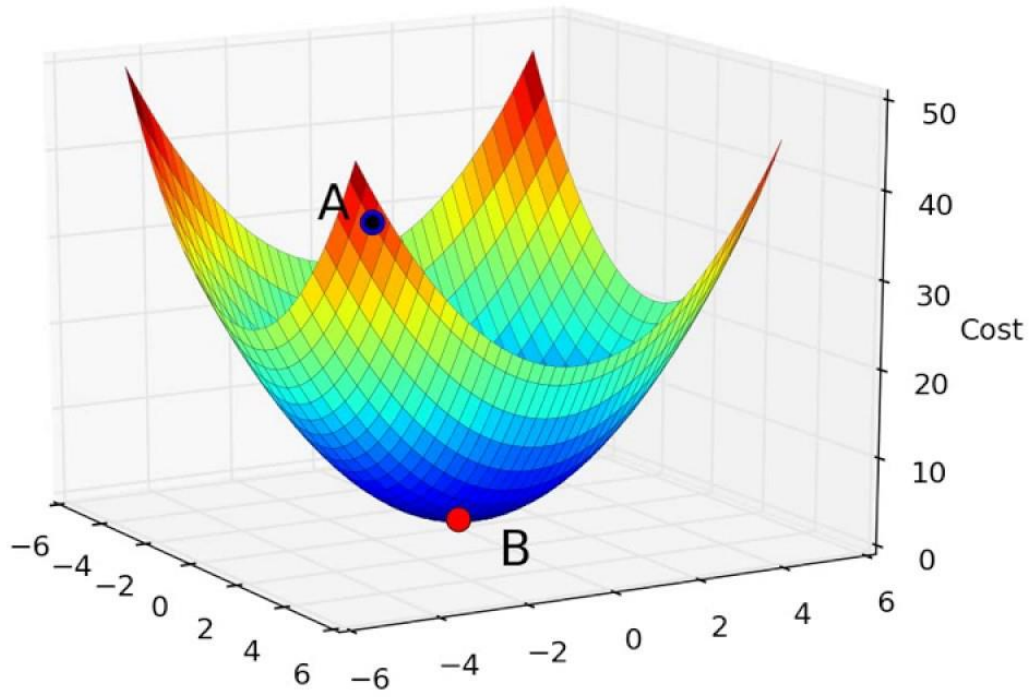
¿QUÉ ES CUDA?

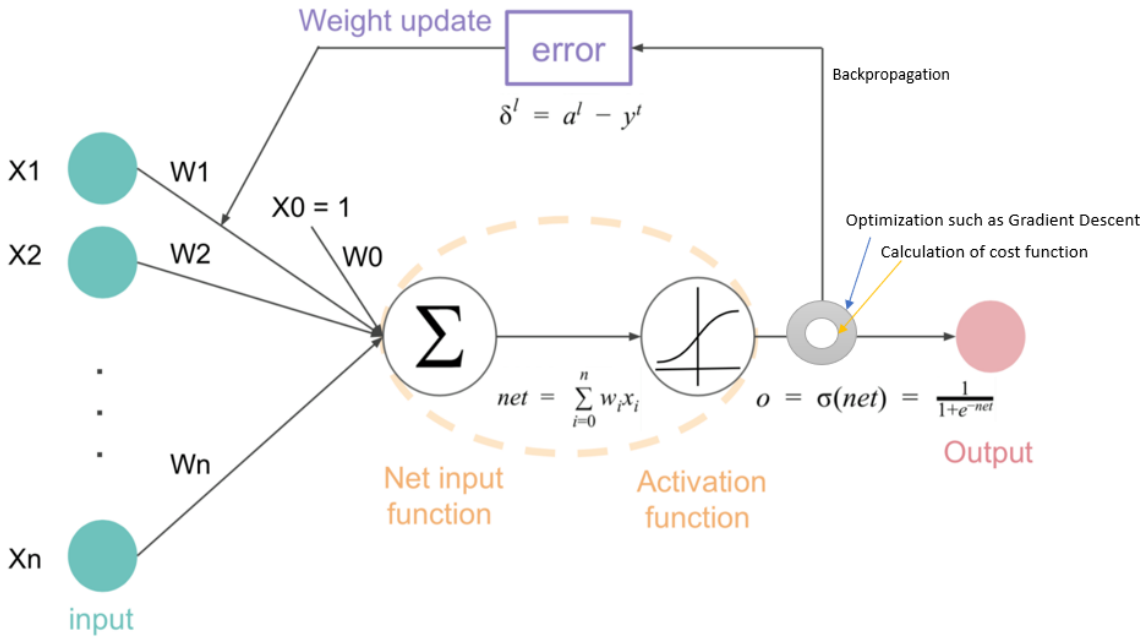
PROTEIN FOLDING



¿QUÉ ES CUDA?

GRADIENT DESCENT Y BACKPROPAGATION





¿QUÉ ES CUDA?

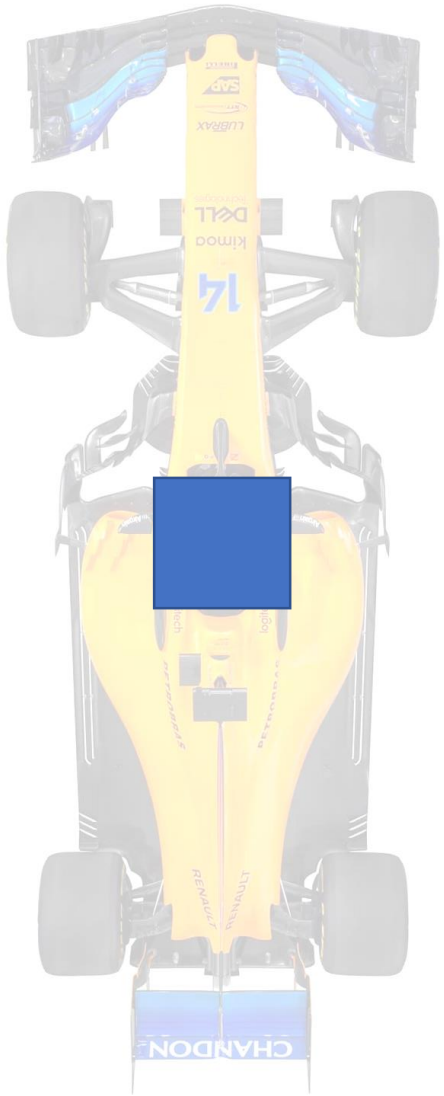
NVIDIA GEFORCE 8800 GTX (2007)

¿QUÉ ES CUDA?

ANALOGÍA

¿QUÉ ES CUDA?

ANALOGÍA

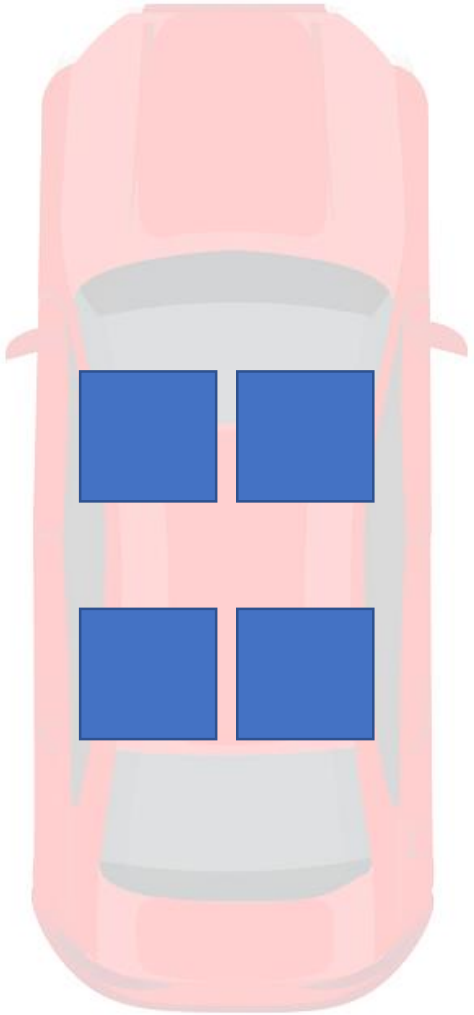


¿QUÉ ES CUDA?

ANALOGÍA

¿QUÉ ES CUDA?

ANALOGÍA

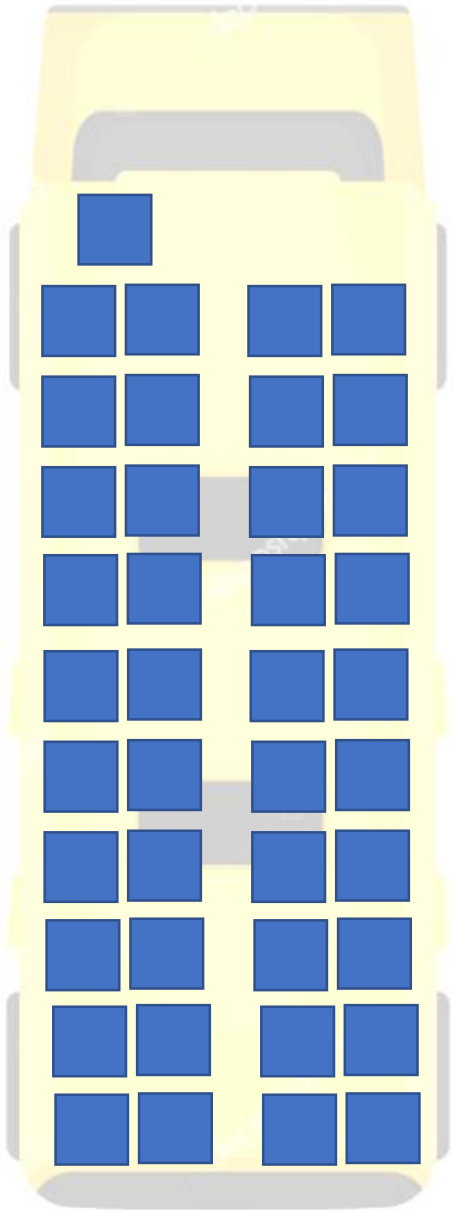


¿QUÉ ES CUDA?

ANALOGÍA

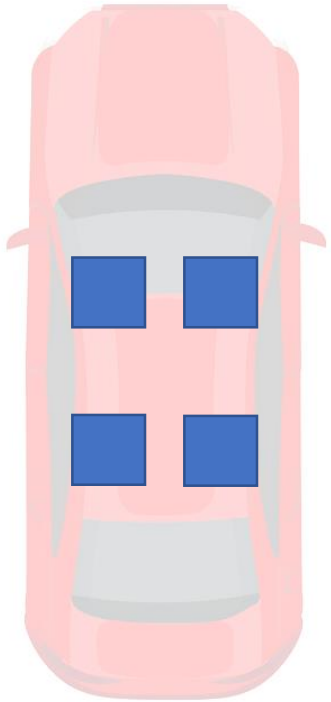
¿QUÉ ES CUDA?

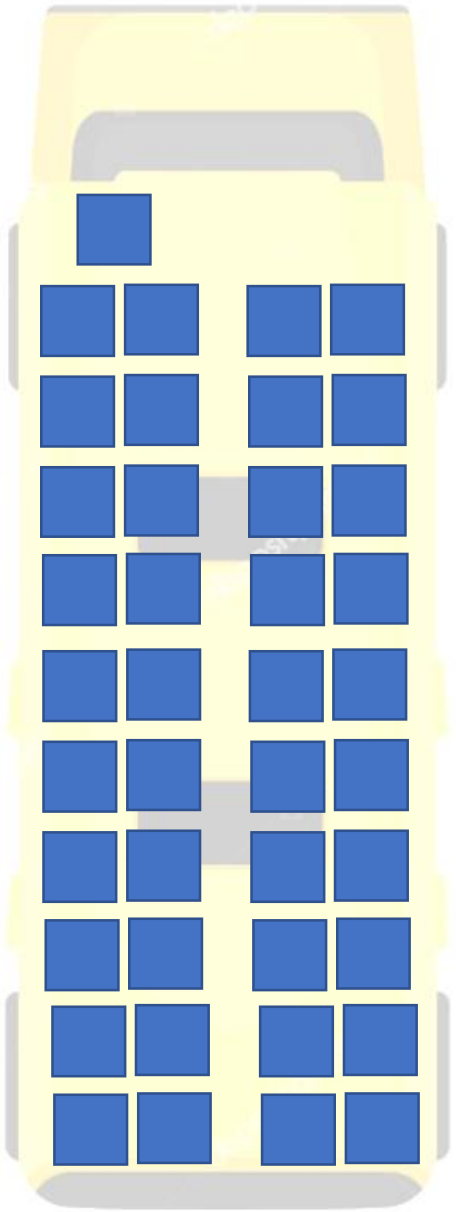
ANALOGÍA



¿QUÉ ES CUDA?

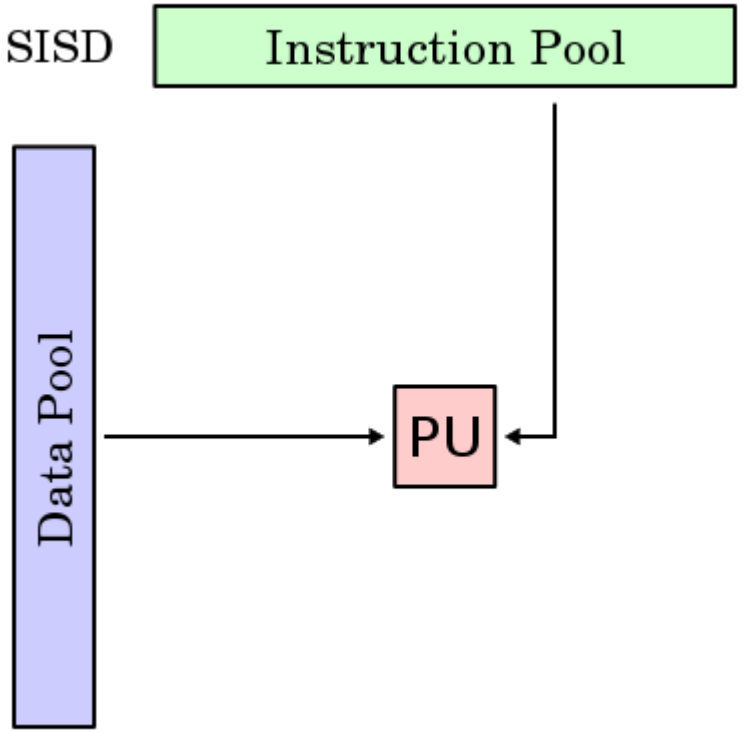
ANALOGÍA





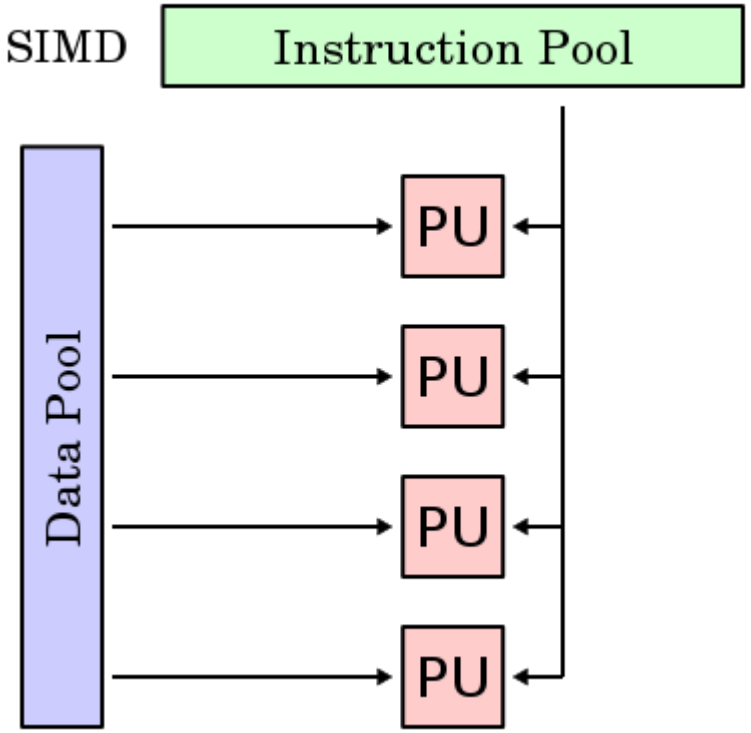
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (SISD)



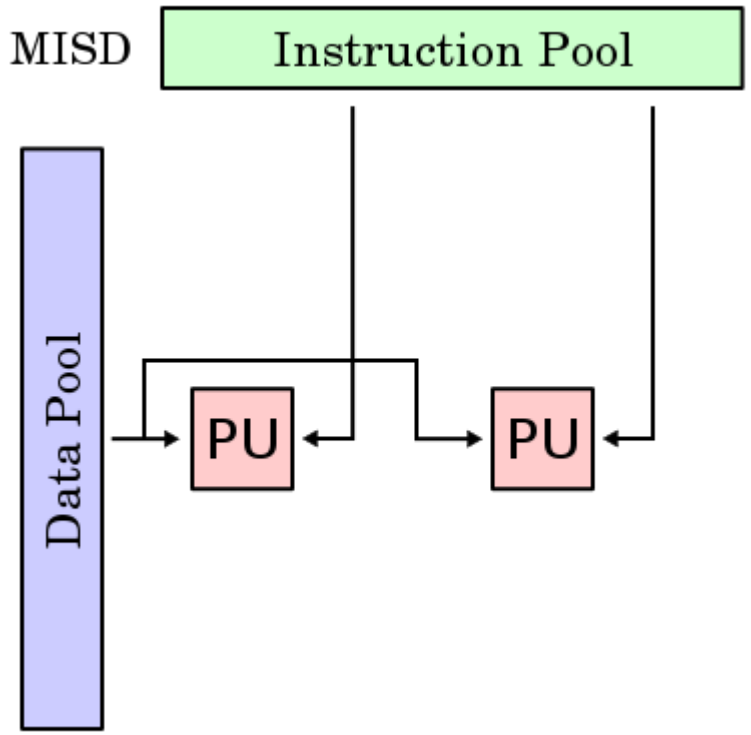
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (SIMD)



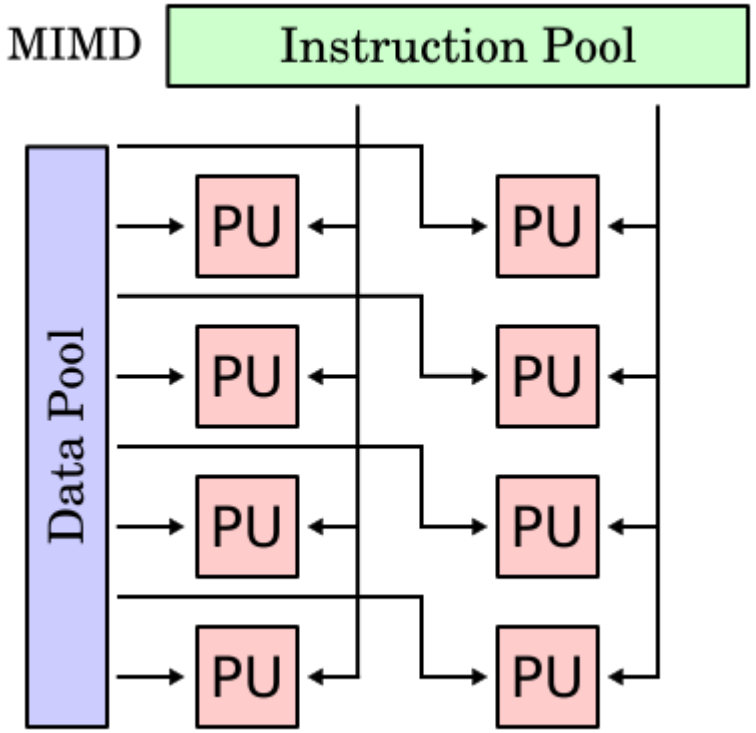
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (MISD)



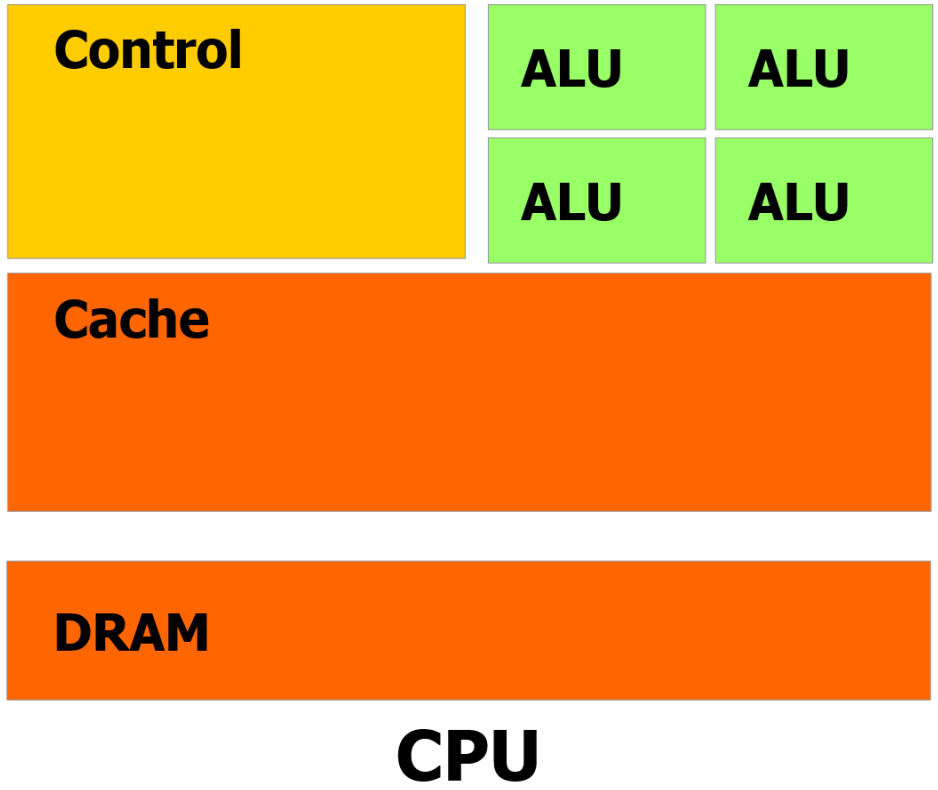
ARQUITECTURA HARDWARE

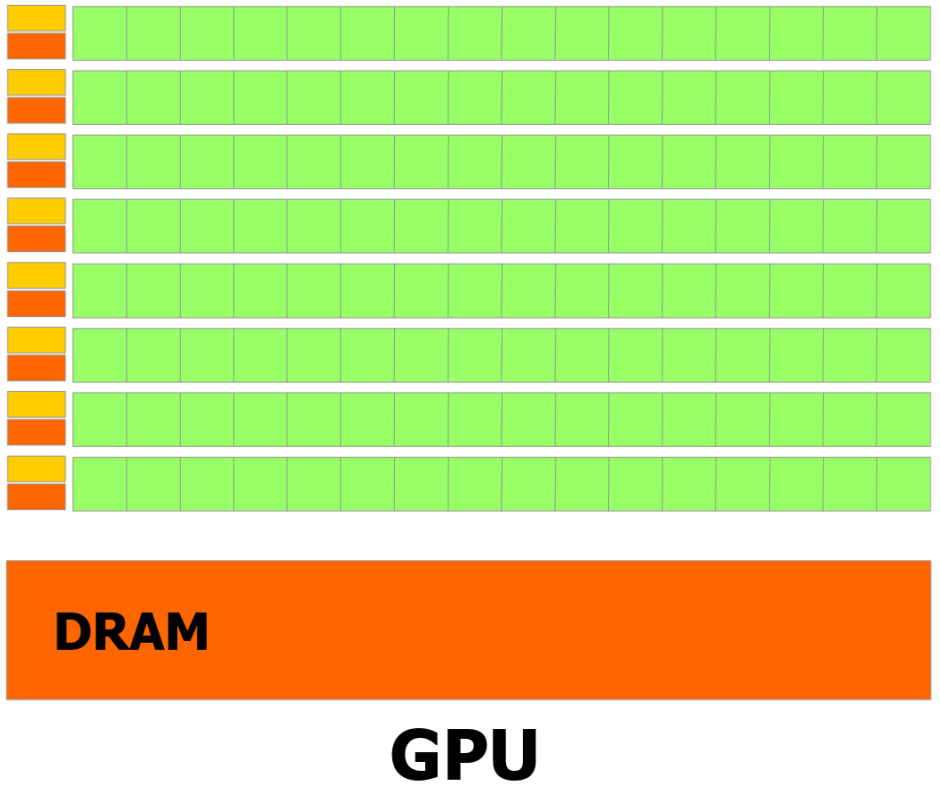
TAXONOMÍA DE FLYNN (MIMD)



ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU



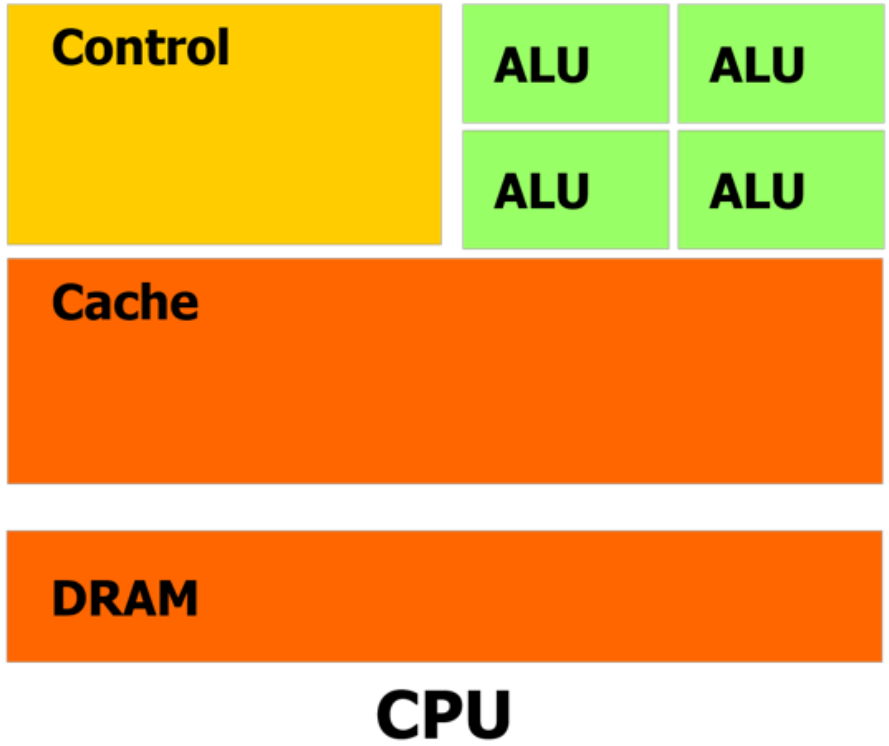


ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU

CPU

- **Caches grandes**
  - Permiten bajar la latencia en los accesos a memoria.
- **Unidad de control compleja**
  - Branch prediction
  - Data forwarding
- **ALUs complejas**
  - Reducen latencia de las operaciones

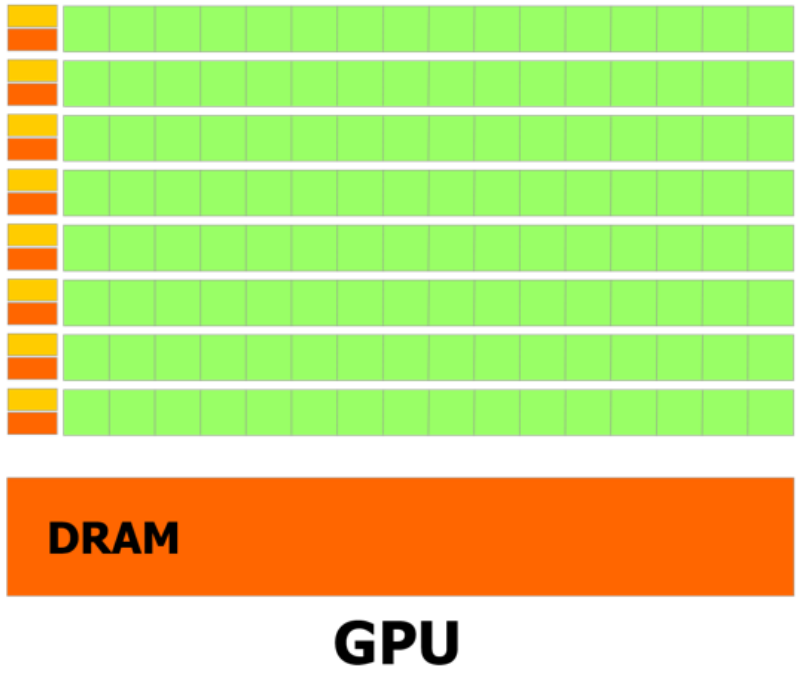


ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU

GPU

- **Caches pequeñas**
  - Potenciar ancho banda memoria.
- **Unidad de control simple**
  - No Branch prediction
  - No Data forwarding
- **Muchas ALUs simples**
  - Altamente segmentadas para aumentar el ancho de banda de la memoria
- **Requiere un número muy elevado de hilos para ocultar las latencias**



ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU

- **Los hilos en la GPU son muy ligeros.**
  - Se necesita muy poco tiempo para crear/destruir hilos.
- **La GPU necesita muchos hilos para ser eficiente.**
  - Una CPU multi-núcleo solo necesita unos pocos hilos.
- **El tiempo de acceso a memoria en la GPU es alto.**
  - En la CPU el ancho de banda es menor.

ARQUITECTURA HARDWARE

COMPARACIÓN ENTRE CPU Y GPU

- **Intel Core i7**
  - 4-8 núcleos MIMD
  - Pocos registros, cache multi-nivel
  - **10-30 GB/s** ancho de banda hacia la memoria principal

- **NVIDIA GTX480**
  - 512 núcleos, organizados en 16 unidades SM cada una con 32 núcleos.
  - Muchos registros, inclusión cachés nivel 1 y 2.
  - 5 GB/s ancho de banda hacia el procesador HOST.
  - **180 GB/s** ancho de banda memoria tarjeta gráfica.

ARQUITECTURA HARDWARE

COMPONENTES BÁSICOS DE UNA GPU CUDA

INTERFAZ CON EL HOST (CPU) QUE CONECTA LA GPU AL BUS PCI EXPRESS

0-2 COPY ENGINES PARA TRANSFERENCIAS DE MEMORIA ASÍNCRONAS

INTERFAZ DE MEMORIA DRAM QUE CONECTA A LA GPU CON SU MEMORIA INTERNA

CIERTO NÚMERO DE GRAPHICS PROCESSING CLUSTERS (GPCS) CON STREAMING MULTIPROCESSORS (SMS) O PROCESADORES DE GENERACIÓN ACTUAL

ARQUITECTURA HARDWARE

STREAMING MULTIPROCESSORS

INSTRUCTION CACHE/DECODER

SCHEDULER

CUDA CORES

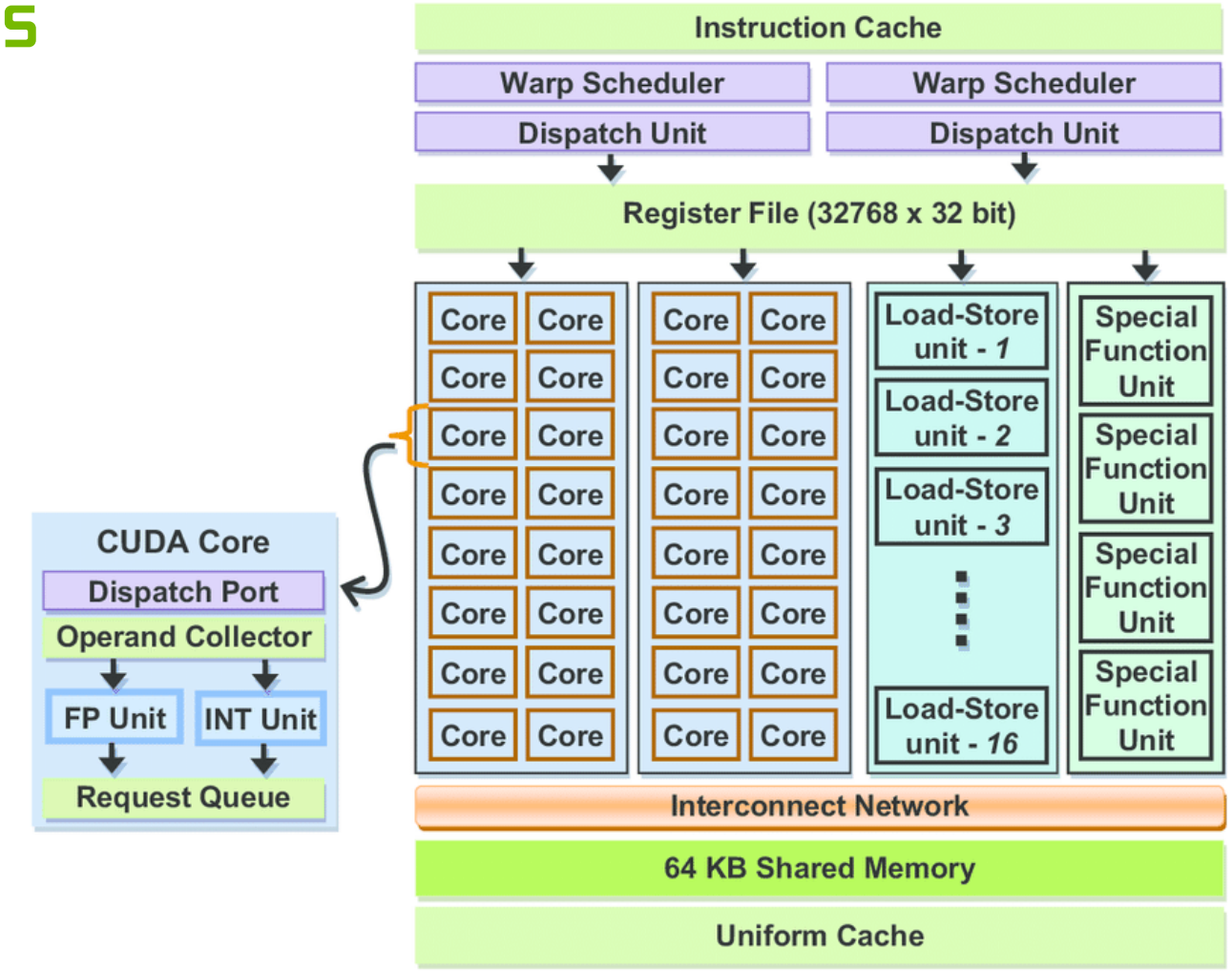
UNIDADES LOAD-STORE

UNIDADES SFU

MEMORIA COMPARTIDA

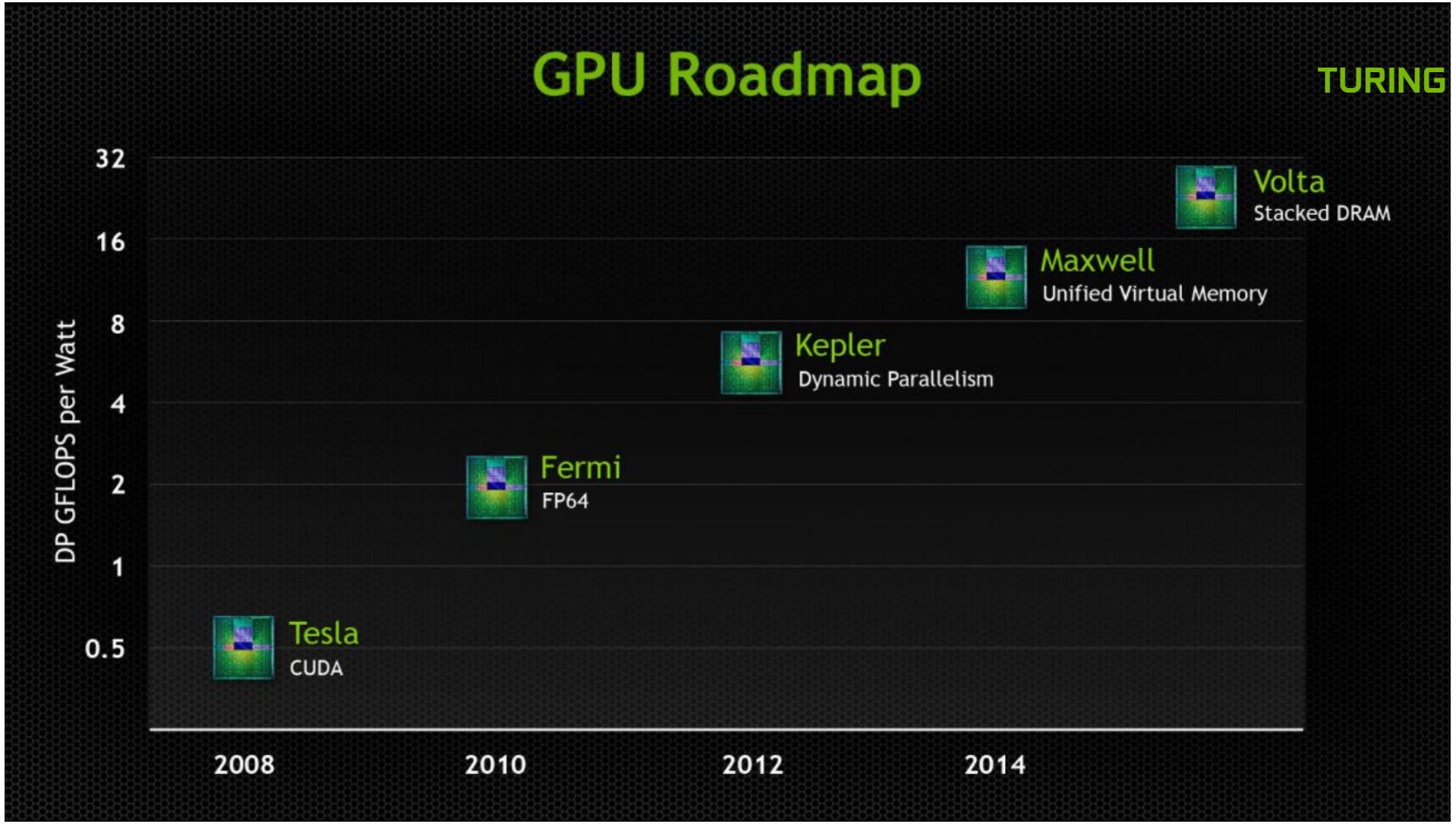
CACHÉS DE TEXTURAS

ARCHIVO DE REGISTROS



ARQUITECTURA HARDWARE

EVOLUCIÓN DE MICROARQUITECTURAS



ARQUITECTURA HARDWARE

GPU TECHNOLOGY CONFERENCE 2018

ARQUITECTURA SOFTWARE

CUDA

- **CUDA (Compute Unified Device Architecture)**
  - Hace referencia tanto a un compilador como a un conjunto de herramientas de desarrollo creadas por NVIDIA.

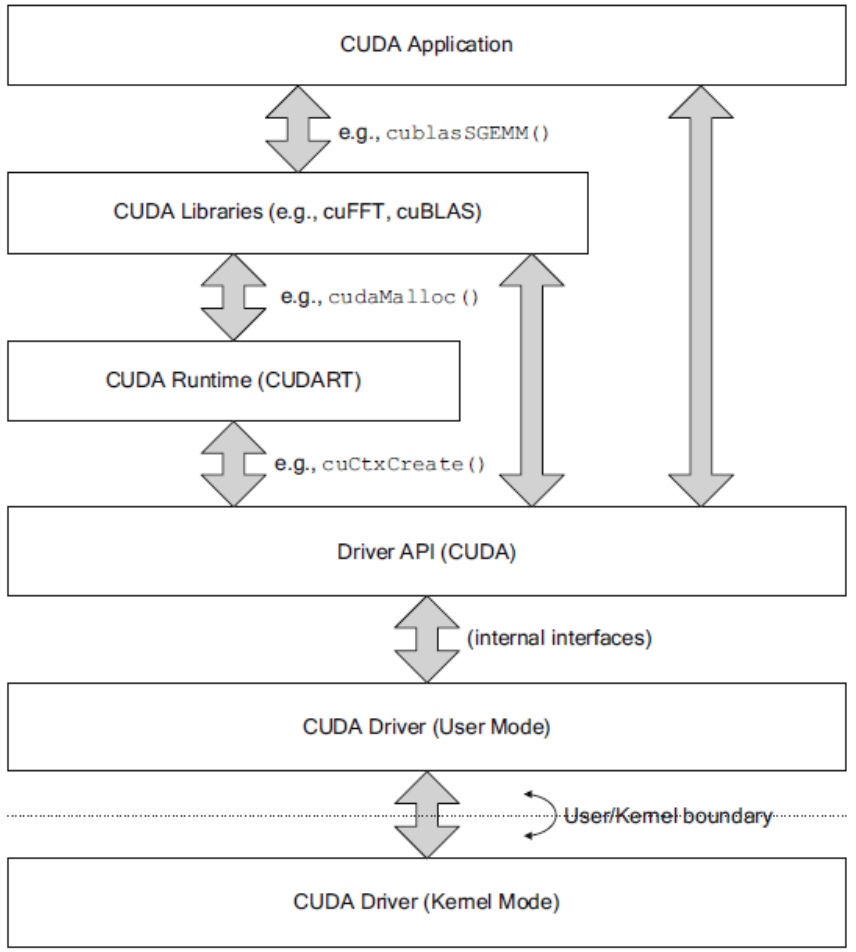
- **Basado en C/C++** con algunas extensiones

- Soporte C++ , Fortran. Wrappers para otros lenguajes: .NET, Python, Java…

- **Gran cantidad de ejemplos** y buena documentación, lo cual reduce la curva de aprendizaje para aquellos con experiencia en lenguajes como OpenMPI y MPI.
- Extensa comunidad de usuarios en los **foros de NVIDIA**.

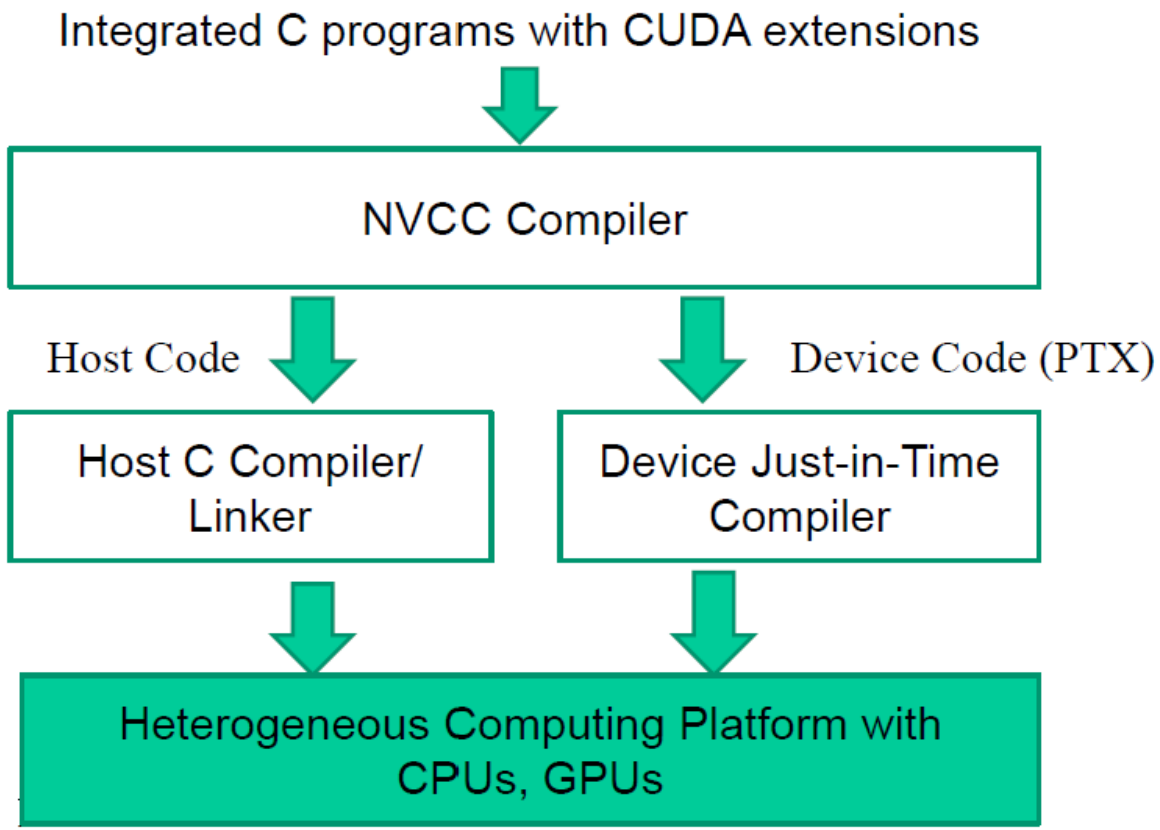
ARQUITECTURA SOFTWARE

STACK CUDA



ARQUITECTURA SOFTWARE

COMPILACIÓN



ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

KERNEL: es una función que al ejecutarse lo hará en una gran cantidad de hilos y en la GPU.

BLOCK: es una agrupación de hilos en 1D, 2D ó 3D. Cada bloque se ejecuta sobre un único SM, pero un SM puede tener asignados varios bloques para ejecución.

GRID: es una forma de estructurar los bloques, bien en 1D, 2D ó 3D

ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

INVOCACIÓN DE KERNEL:

```
kernel_routine<<<grid_dim, block_dim>>> (args...);
```

- **gridDim :** número de bloques. Tamaño del grid.
- **blockDim:** número de hilos que se ejecutan dentro de un bloque.
- **Args:** número limitado de argumentos, normalmente punteros a memoria de la GPU.

TAMAÑOS DE BLOQUE Y MALLA DEFINIDOS CON DIM3:

```
dim3 block_dim (32, 32, 1); // 1024 hilos en bloque de 32 x 32 x 1 (2D)
dim3 grid_dim (4, 4, 1); // 16 bloques en malla de 4 x 4 x 1 (2D)
```

ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

CADA HILO EJECUTA UNA COPIA DEL KERNEL, Y DISPONE DE LA SIGUIENTE INFORMACIÓN:

VARIABLES PASADAS COMO ARGUMENTO

- PUNTEROS A MEMORIA GPU
- VARIABLES SIMPLES POR VALOR (INT, FLOAT, CHAR, LÍMITE DE TAMAÑO)

CONSTANTES GLOBALES EN MEMORIA GPU

VARIABLES ESPECIALES (DIM3) PARA IDENTIFICAR AL HILO:

gridDim (tamaño de la malla)
blockDim (tamaño de los bloques)
blockIdx (identificador de bloque) LOCAL PARA CADA BLOQUE
threadIdx (identificador de hilo) LOCAL PARA CADA BLOQUE

ARQUITECTURA SOFTWARE

FLUJO DE EJECUCIÓN

En el nivel más alto encontramos un proceso sobre la CPU (Host) que realiza los siguientes pasos:

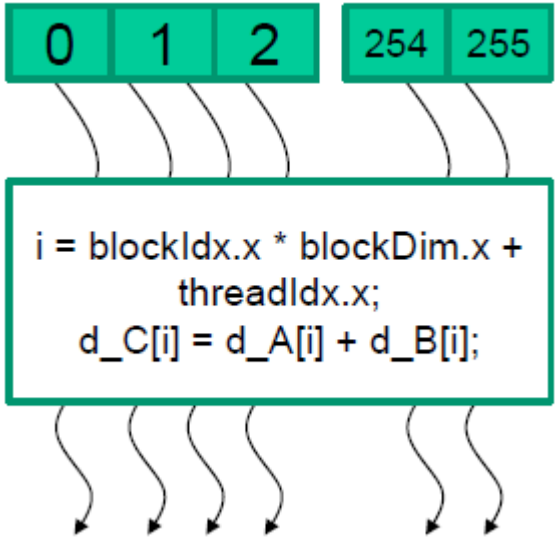
1. Inicializa GPU
2. Reserva memoria en la parte host y device
3. Copia datos desde el **host** hacia la memoria **device**
4. Lanza la ejecución de múltiples copias del **kernel**
5. Copia datos desde la memoria **device** al **host**
6. Se repiten los pasos 3-5 tantas veces como sea necesario
7. Libera memoria y finaliza la ejecución proceso maestro

ARQUITECTURA SOFTWARE

EJECUCIÓN DE UN KERNEL

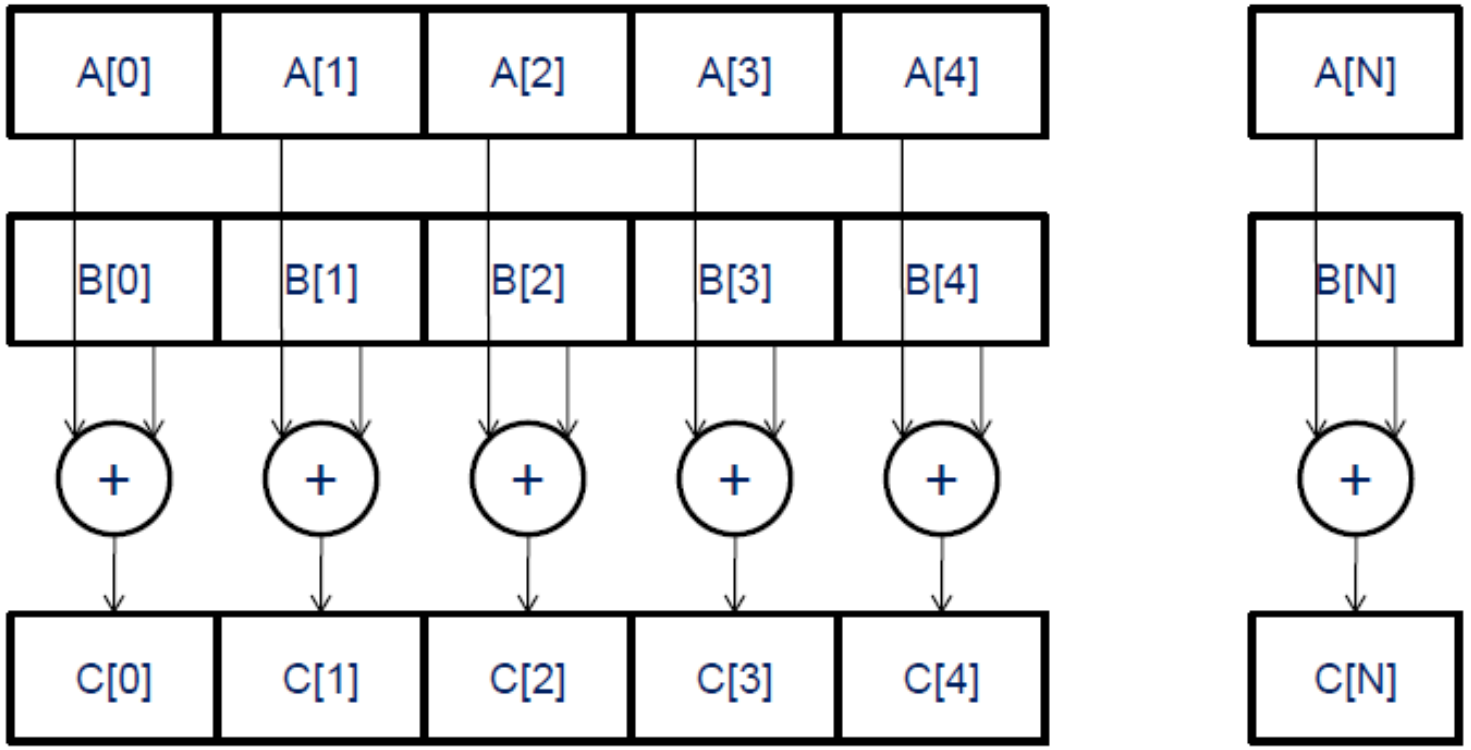
Un kernel CUDA se ejecuta sobre un grid de hilos:

- Todos los threads del grid ejecutan el mismo código
- Cada hilo tiene sus índices y lo utiliza para direccionar la memoria y realizar su lógica.



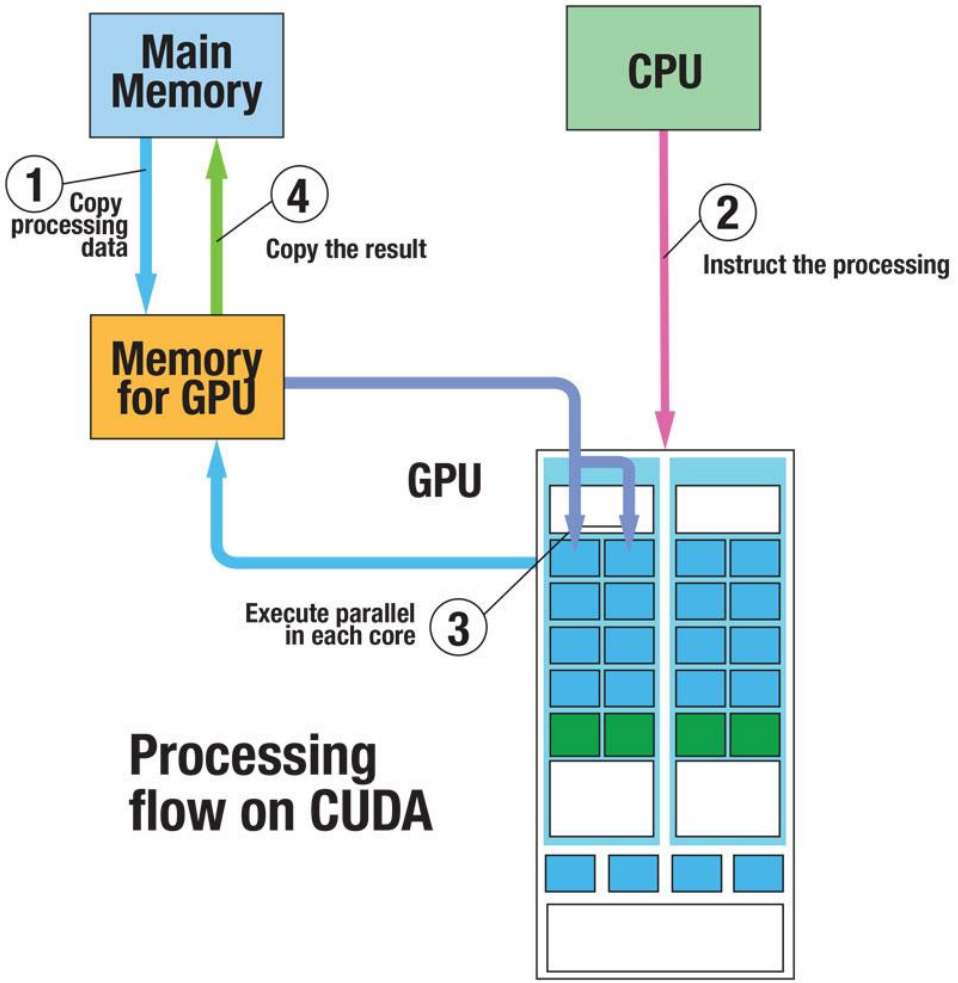
ARQUITECTURA SOFTWARE

EJEMPLO: SUMA DE VECTORES



FLUJO DE EJECUCIÓN

SUBTÍTULO QUE NADIE LEE



COMPUTE CAPABILITY

VENDO FORD FIESTA 1.5TDI CON 80000 KM

Feature support (unlisted features are supported for all compute abilities)	Compute capability (version)											
	1.0	1.1	1.2	1.3	2.x	3.0	3.2	3.5, 3.7, 5.0, 5.2	5.3	6.x	7.x	8.x
Integer atomic functions operating on 32-bit words in global memory	No	Yes										
atomicExch() operating on 32-bit floating point values in global memory												
Integer atomic functions operating on 32-bit words in shared memory	No				Yes							
atomicExch() operating on 32-bit floating point values in shared memory												
Integer atomic functions operating on 64-bit words in global memory												
Warp vote functions												
Double-precision floating-point operations	No				Yes							
Atomic functions operating on 64-bit integer values in shared memory	No					Yes						
Floating-point atomic addition operating on 32-bit words in global and shared memory												
_ballot()												
_threadfence_system()												
_syncthreads_count(), _syncthreads_and(), _syncthreads_or()												
Surface functions												
3D grid of thread block												
Warp shuffle functions	No					Yes						
Funnel shift	No						Yes					
Dynamic parallelism	No							Yes				
Half-precision floating-point operations: addition, subtraction, multiplication, comparison, warp shuffle functions, conversion	No								Yes			
Atomic addition operating on 64-bit floating point values in global memory and shared memory	No									Yes		
Tensor core	No										Yes	

https://en.wikipedia.org/wiki/CUDA#Version_features_and_specifications

DEVICE QUERY

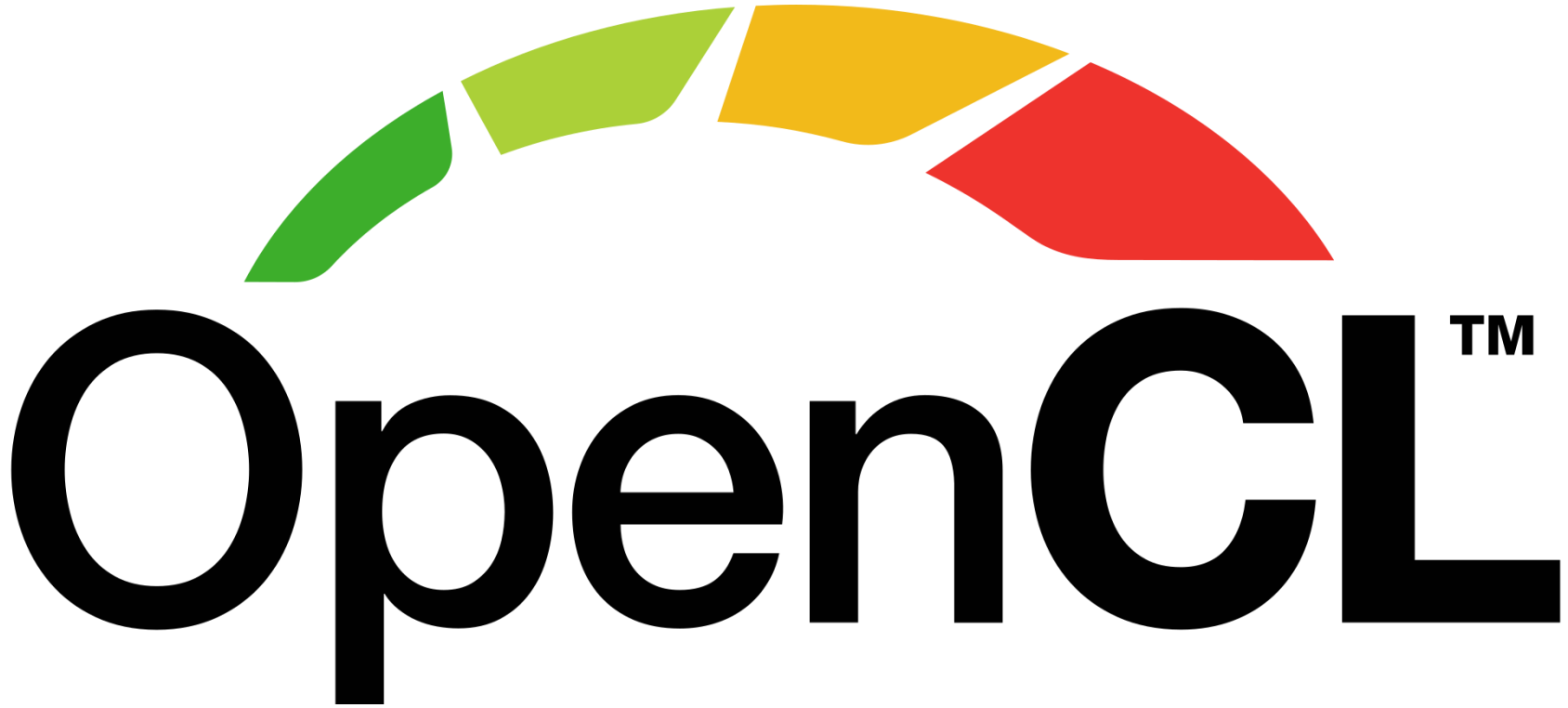
CONOCER DE FORMA PROGRAMÁTICA LAS CARACTERÍSTICAS DE LA GPU

C:\ProgramData\NVIDIA Corporation\CUDA Samples\v8.0\1_Utilities\deviceQuery\../../bin/win64/Debug/deviceQuery.exe

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v8.0\1_Utilities\deviceQuery\../../bin/win64/Debug/deviceQuery.exe Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX TITAN X"
  CUDA Driver Version / Runtime Version          8.0 / 8.0
  CUDA Capability Major/Minor version number:    5.2
  Total amount of global memory:                 12288 MBytes (12884901888 bytes)
  (24) Multiprocessors, (128) CUDA Cores/MP:     3072 CUDA Cores
  GPU Max Clock rate:                            1076 MHz (1.08 GHz)
  Memory Clock rate:                             3505 Mhz
  Memory Bus Width:                              384-bit
  L2 Cache Size:                                 3145728 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     Yes
```

ALTERNATIVAS

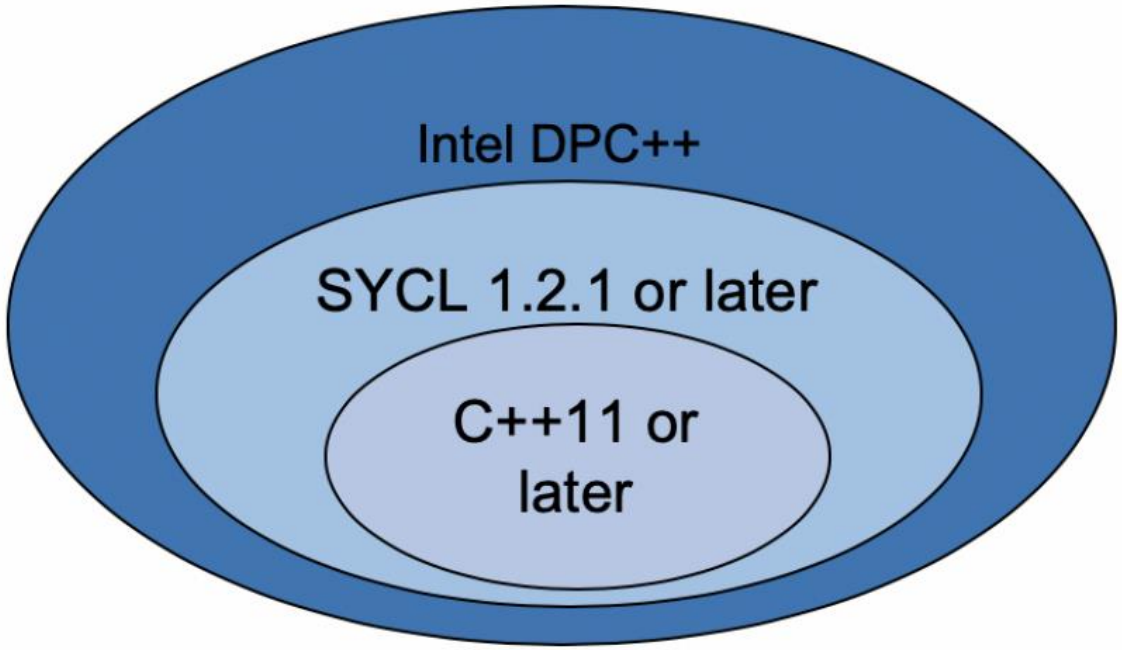
OTROS MODELOS DE COMPUTACIÓN HETEROGÉNEA



ALTERNATIVAS

OTROS MODELOS DE COMPUTACIÓN HETEROGÉNEA

SYCL™



INTRODUCCIÓN A CUDA

X JORNADAS DE DIVULGACIÓN DE APLICACIONES CIENTÍFICAS Y VISIÓN POR COMPUTADOR SOBRE PROCESADORES GRÁFICOS

UNIVERSIDAD DE ALICANTE

Pablo Martínez-González <pmartinez@dtic.ua.es>

Albert García-García <agarcia@dtic.ua.es>

José García-Rodríguez <jgarcia@dtic.ua.es>